

Encadré par :

Pr. AIT BENNACER Sara

Réalisé par :

Nom & Prénom

Rapport Mini-projet

Sujet

**Table des matières**

[**1.** **Objectifs et contexte :** 2](#_Toc135581428)

[**2.** **Méthodologie :** 2](#_Toc135581429)

[**3.** **Résultats :** 2](#_Toc135581430)

[**4.** **Analyse :** 2](#_Toc135581431)

[**5.** **Conclusion :** 2](#_Toc135581432)

[**6.** **Bibliographie :** 2](#_Toc135581433)

**Liste des Figures**

[**1.** **Figure 1 :** 2](#_Toc135581428)

# **Objectifs et contexte :**

Contexte du projet :

On va réalisé un projet pour la Création d'une application de gestion location des Voitures :

Une agence de location de voitures souhaite avoir une application en

informatique pour gérer son activité, L’agence possède un parc de voitures de marques différentes, 0.5dh par kilomètre est le prix minimal défini par l’agence pour la location d’une voiture.

L’application demandée se base sur un historique pour chaque voiture:

 Marque

 Matricule

 Kilométrage parcouru

 Disponibilité (disponible dans le parc ou louée)

À chaque demande de location, on doit vérifier d’abord la disponibilité, et

afficher les informations sur les voitures disponibles avec leurs (marque,

matricule...) « les informations doivent être affichées triées dans un ordre

croissant selon le kilométrage parcouru. Si le client a ciblé une marque

précise, l’affichage des informations ne doit concerner que cette marque.

Une fois l’opération de location est faite : actualiser la base d’information

« disponibilité »

A chaque rendement d’une voiture, ses informations dans la base doivent être

actualisées

Disponibilité, kilométrage et on doit calculer et afficher le prix à payer par le

client.

Objectifs :

On recherche à réaliser les opérations suivantes :

1. Gestion des Voitures

2. Ajouter une nouvelle voiture (Marque, matricule, kilométrage,

Disponibilité)

3. Afficher la marque, le Matricule, le kilométrage et la disponibilité

D’une voiture donnée,

4. Modifier la marque, le Matricule, le kilométrage et la disponibilité

D’une voiture donnée,

5. Supprimer la marque, le Matricule, le kilométrage et la disponibilité

D’une voiture donnée,

N.B. : Le matricule de la voiture doit être unique.

6. Gestion des Locations

- Ajouter un nouveau Client qui va louer une voiture existante et

disponible.

Le Client doit avoir les informations suivantes :

 Le Nom,

 le CIN,

 le Téléphone,

 le Matricule de la voiture désirée,

 l’avance,

 le Prix a payé,

- Afficher le Client avec le reste d’informations dont le CIN donné,

- Modifier le Client avec ces informations dont le CIN donné,

- Supprimer le client dont le CIN donné,

Importance du projet :

Le programme propose une solution concrète à un problème courant, à savoir la gestion des locations de voitures. Il offre des fonctionnalités utiles pour les agences de location ou toute autre entreprise opérant dans le domaine de la location de voitures le mini-projet utilise des structures de données pour stocker les informations relatives aux voitures et aux clients. Cette approche permet de gérer efficacement les données, de les organiser de manière structurée et d'accéder facilement aux informations nécessaires et il offre des fonctionnalités pour ajouter, modifier, afficher et supprimer des voitures et des clients. Il permet également de rechercher des voitures par leur matricule ou des clients par leur numéro de CIN. Ces fonctionnalités facilitent la manipulation et la gestion des données, en offrant des opérations courantes sur les objets du système Le programme propose un menu interactif pour permettre à l'utilisateur d'interagir avec le système. L'utilisateur peut effectuer différentes actions en choisissant parmi les options du menu. Cela rend le programme convivial et facile à utiliser pour les personnes chargées de la gestion des locations de voitures Le programme peut être étendu pour inclure de nouvelles fonctionnalités ou des améliorations supplémentaires en fonction des besoins spécifiques de l'entreprise ou de l'utilisateur. Il sert de base solide pour développer un système de gestion de location de voitures plus complet et évolutif**.**

Objectif principal du projet :

**L'importance de ce mini-projet réside dans sa capacité à fournir une base solide pour développer un système de gestion de location de voitures plus complet et adapté aux besoins spécifiques d'une agence de location ou d'une entreprise similaire. Il permet également aux utilisateurs de comprendre les concepts de base de la programmation structurée, tels que l'utilisation de structures, les boucles, les fonctions, et l'interaction avec l'utilisateur via une interface simple**

# **Méthodologie :**

La méthodologie utilisée dans ce code pour mener à bien le projet de gestion de location de voitures est la suivante :

Définition des structures de données :

**struct voiture** : contient les informations sur une voiture (marque, matricule, kilométrage, disponibilité).

**struct Client** : contient les informations sur un client (nom, CIN, téléphone, matricule de voiture désirée, avance, prix).

Déclaration des tableaux pour stocker les voitures et les clients :

**voiture[MAX\_voiture]** : tableau de structures pour stocker les voitures.

**clients[MAX\_CLIENTS]** : tableau de structures pour stocker les clients.

Implémentation des fonctions pour manipuler les voitures :

**ajoutervoiture()** : permet d'ajouter une nouvelle voiture en demandant à l'utilisateur de saisir les informations.

**affichervoiture(const struct voiture \*voiture)** : affiche les informations d'une voiture passée en paramètre.

**afficherinformation()** : affiche la liste des voitures en appelant la fonction **affichervoiture** pour chaque voiture.

**findCarByRegistration(const char \*registration)** : recherche une voiture par son matricule et renvoie un pointeur vers la voiture trouvée.

**modifiervoiture()** : permet de modifier les informations d'une voiture existante en demandant à l'utilisateur de saisir les nouvelles informations.

**effacervoiture()** : supprime une voiture du tableau en demandant à l'utilisateur de saisir le matricule de la voiture à supprimer.

Implémentation des fonctions pour manipuler les clients :

**ajouterClient()** : permet d'ajouter un nouveau client en demandant à l'utilisateur de saisir les informations.

**AIC(const struct Client \*client)** : affiche les informations d'un client passé en paramètre.

**infoClients()** : affiche la liste des clients en appelant la fonction **AIC** pour chaque client.

**trouverClientavecCIN(const char \*cin)** : recherche un client par son CIN et renvoie un pointeur vers le client trouvé.

**modifierClient()** : permet de modifier les informations d'un client existant en demandant à l'utilisateur de saisir les nouvelles informations.

**effacerClient()** : supprime un client du tableau en demandant à l'utilisateur de saisir le CIN du client à supprimer.

Implémentation de la fonction **main()** :

Affichage du menu principal et récupération du choix de l'utilisateur.

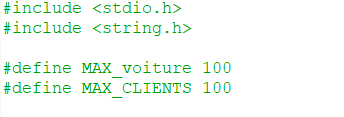
En fonction du choix, affichage des sous-menus pour la gestion des voitures ou des locations.

Dans chaque sous-menu, appel des fonctions correspondantes en fonction du choix de l'utilisateur.

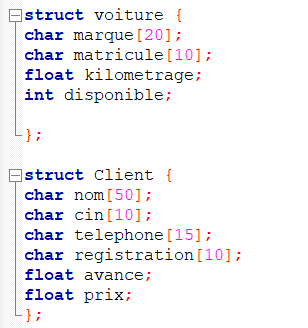
Le programme se répète jusqu'à ce que l'utilisateur choisisse l'option "Quitter" (0).

Ce code est un programme de gestion de location de voitures. Voici les différentes étapes qu'il suit :

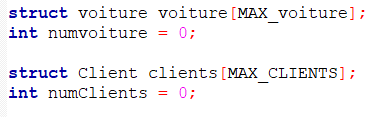
1-Définition des constantes MAX\_voiture et MAX\_CLIENTS pour déterminer la taille maximale des tableaux de voitures et de clients respectivement



2-Définition de deux structures de données : "voiture" et "Client", qui représentent respectivement une voiture et un client.

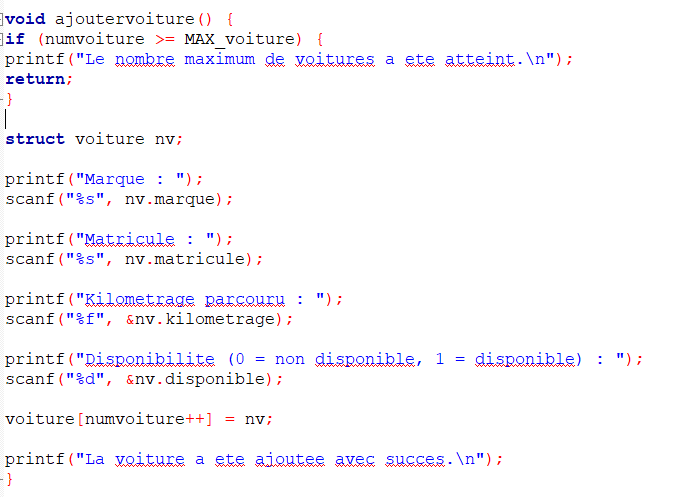


3-Déclaration des tableaux de voitures et de clients : "voiture[MAX\_voiture]" et "clients[MAX\_CLIENTS]".

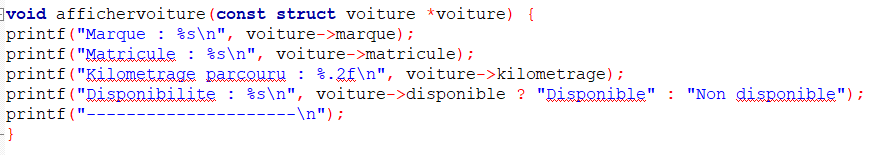


4-Les variables "numvoiture" et "numClients" sont utilisées pour suivre le nombre actuel de voitures et de clients respectivement.

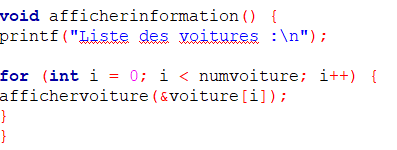
5-La fonction "ajoutervoiture()" permet d'ajouter une nouvelle voiture. Elle vérifie d'abord si le nombre maximal de voitures a été atteint. Si ce n'est pas le cas, elle demande à l'utilisateur de saisir les détails de la nouvelle voiture (marque, matricule, kilométrage et disponibilité) et les stocke dans le tableau des voitures.



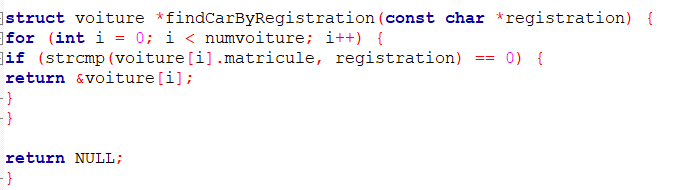
6-La fonction "affichervoiture()" affiche les informations d'une voiture passée en paramètre.



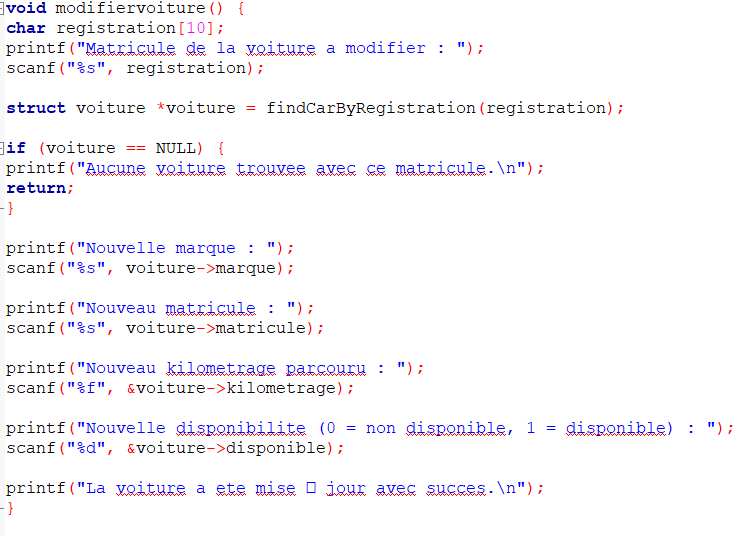
7-La fonction "afficherinformation()" affiche la liste de toutes les voitures en utilisant la fonction "affichervoiture()" pour chaque voiture.



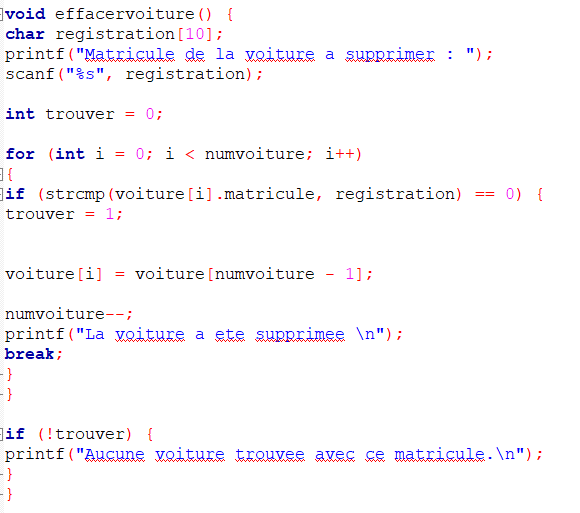
8-La fonction "findCarByRegistration()" recherche une voiture par son matricule dans le tableau des voitures et renvoie un pointeur vers la voiture correspondante. Si aucune voiture n'est trouvée, elle renvoie NULL.



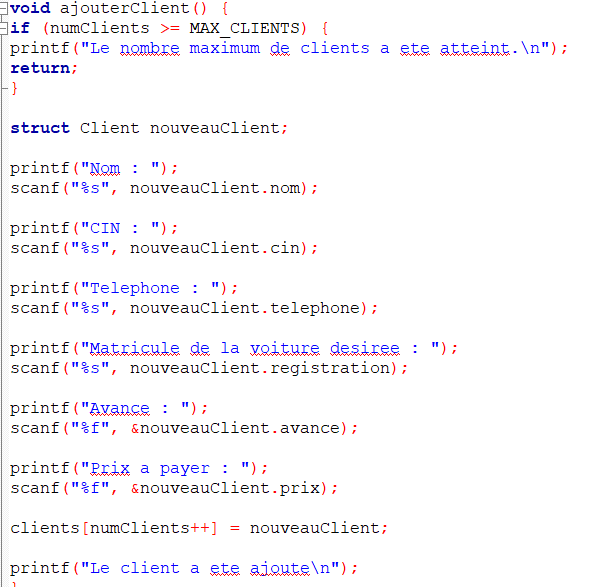
9- La fonction "modifiervoiture()" permet de modifier les informations d'une voiture. Elle demande d'abord le matricule de la voiture à modifier, puis utilise la fonction "findCarByRegistration()" pour obtenir un pointeur vers la voiture correspondante. Ensuite, elle demande à l'utilisateur de saisir les nouvelles informations de la voiture et les met à jour.

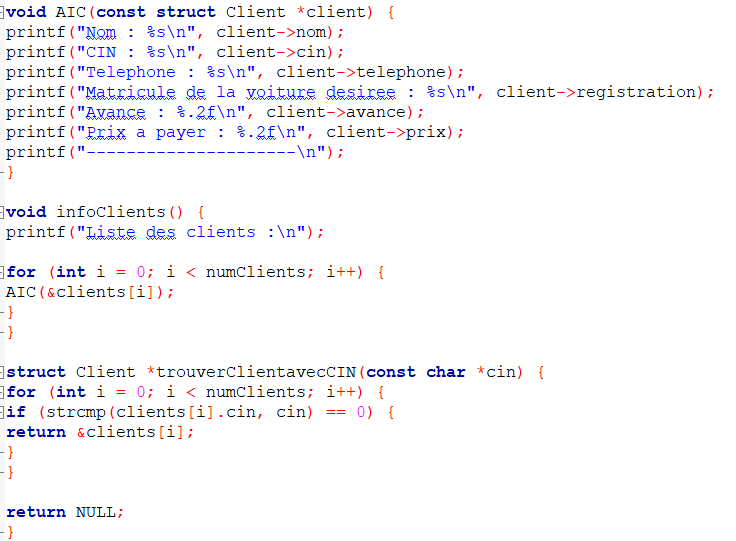


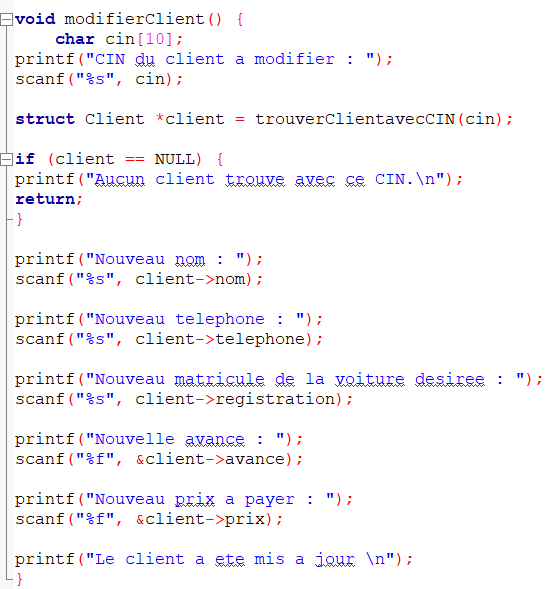
10-La fonction "effacervoiture()" permet de supprimer une voiture du tableau. Elle demande d'abord le matricule de la voiture à supprimer, puis recherche la voiture correspondante à l'aide de la fonction "findCarByRegistration()". Si la voiture est trouvée, elle est remplacée par la dernière voiture du tableau et le nombre de voitures est décrémenté.

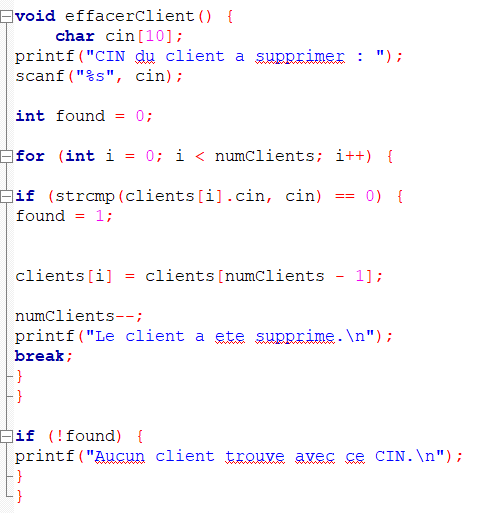


11-Les fonctions "ajouterClient()", "AIC()", "infoClients()", "trouverClientavecCIN()", "modifierClient()" et "effacerClient()" sont similaires aux fonctions précédentes, mais elles s'appliquent aux clients plutôt qu'aux voitures. Elles permettent d'ajouter un nouveau client, d'afficher les informations d'un client, de rechercher un client par son CIN, de modifier les informations d'un client et de supprimer un client.

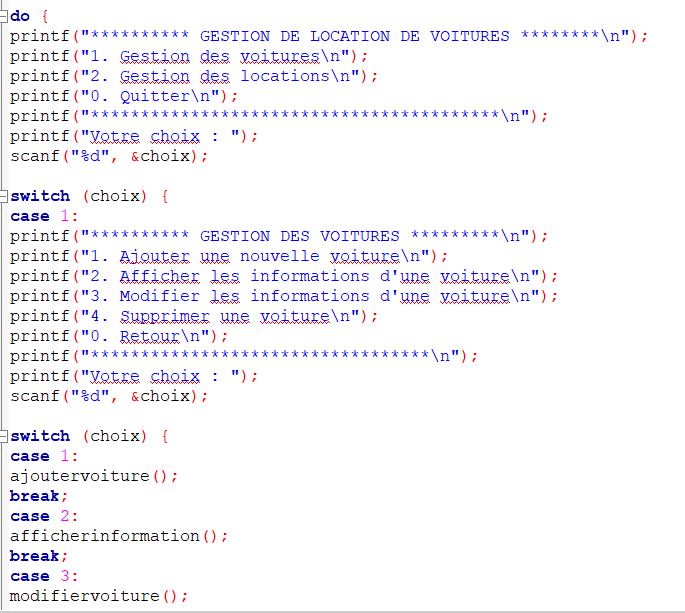


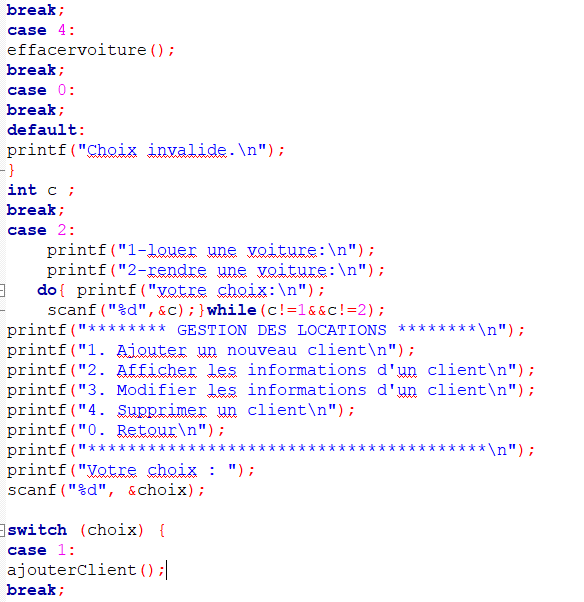


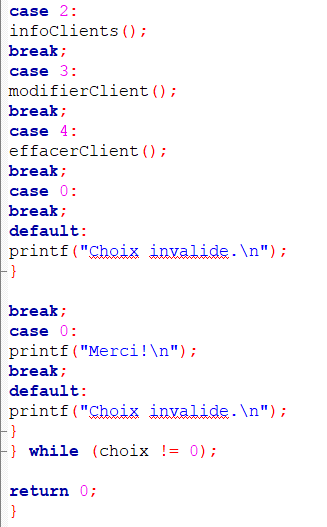




12-La fonction "main()" est le point d'entrée du programme. Elle présente un menu à l'utilisateur pour choisir entre la gestion des voitures, la gestion des locations ou quitter le programme. Selon le choix de l'utilisateur, elle exécute les actions correspondantes en appelant les fonctions appropriées.



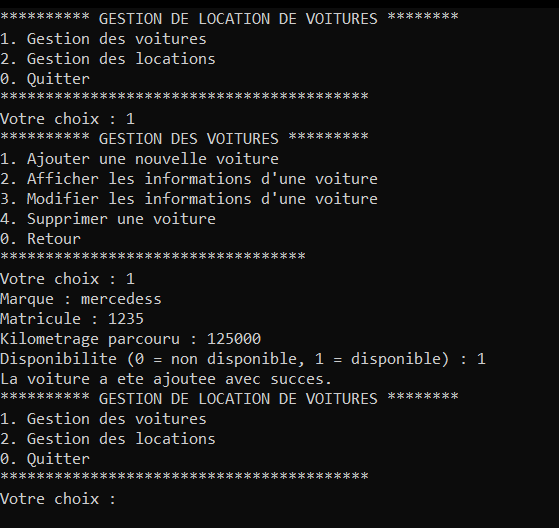




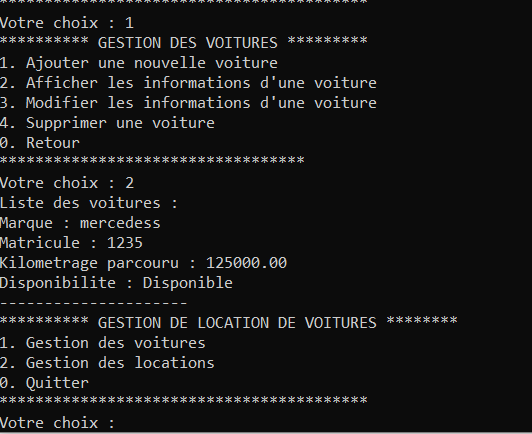
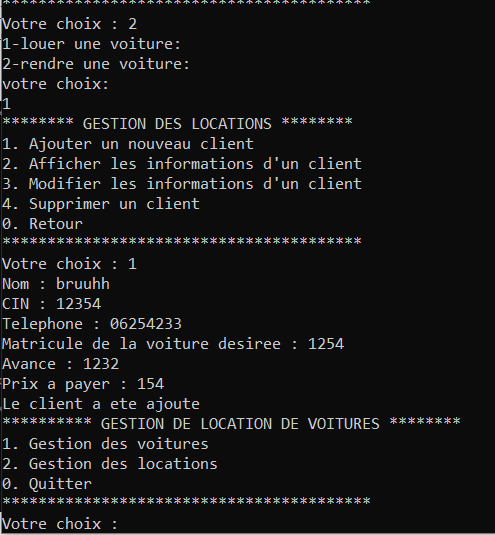
13- Le programme s'exécute en boucle jusqu'à ce que l'utilisateur choisisse de le quitter en entrant "0"

# **Résultats :**

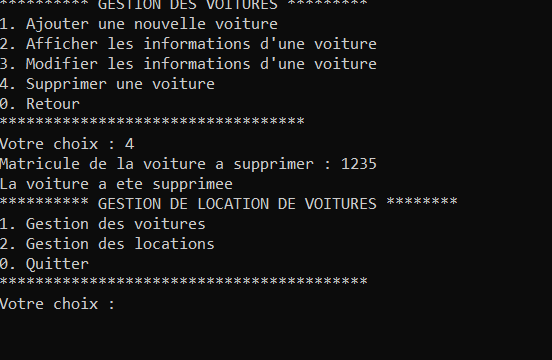
1-ajouter une voiture



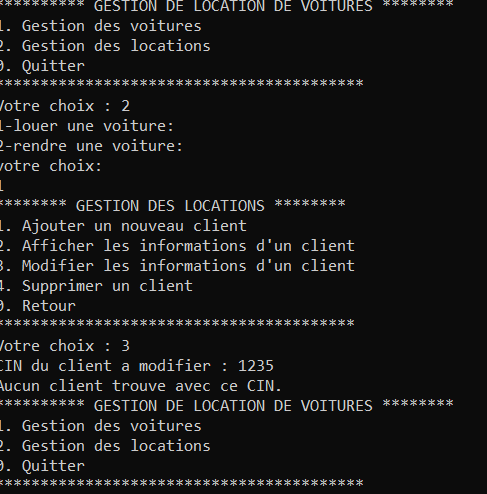
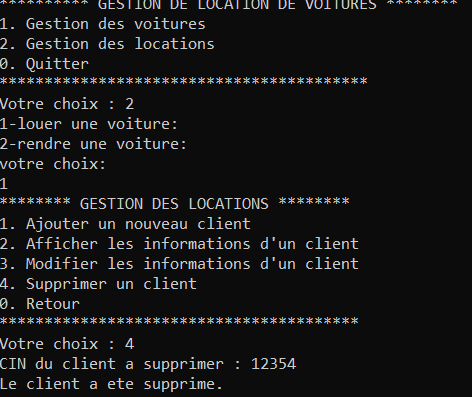
2-ajouter client 3-afficher information voiture



4-suprimer voiture



5-suprimer un client 6-modifier information client



8-modifier information client

# **Analyse :**

Le code utilise des structures pour représenter les voitures et les clients. Les informations sur les voitures sont stockées dans un tableau de structures "voiture", et les informations sur les clients sont stockées dans un tableau de structures "Client". Les différentes fonctions permettent d'effectuer des opérations sur ces données, en utilisant des fonctions auxiliaires pour rechercher des voitures ou des clients spécifiques.

* Evaluez les avantages et les limites de votre méthodologie, et discutez des conclusions que vous pouvez en tirer.

# **Conclusion :**

* Terminez votre rapport en résumant les résultats et en fournissant des
* Recommandations pour l'avenir. Discutez des implications de vos résultats et de leur pertinence pour les futurs projets.

# **Bibliographie :**

* N'oubliez pas de fournir une bibliographie complète des sources que vous avez utilisées es pour mener votre projet.

**Ajout d'une partie de code :** Il est nécessaire d'inclure une section de code dans le rapport. Dans cette section, vous devez expliquer les différentes parties du code ainsi que leur fonctionnement. Vous pouvez donner des instructions sur la façon d'écrire du code de qualité, comme l'utilisation de noms de variables explicites, l'ajout de commentaires pour expliquer les parties complexes du code, etc